# MILES
# Eine Programmierumgebung für die Induktive Logische Programmierung*

*Irene Stahl* *Birgit Tausend*

Institut für Informatik, Universität Stuttgart, Breitwiesenstr. 20-22, D-70565 Stuttgart

stahl@informatik.uni-stuttgart.de tausend@informatik.uni-stuttgart.de

## 1 Einführung

Die *Induktive Logische Programmierung (ILP)* ist ein Forschungsgebiet, das Techniken aus dem Maschinellen Lernen und der Logischen Programmierung vereint. Sie untersucht das klassische Problem induktiven Lernens aus klassifizierten Beispielen im Rahmen der Hornlogik erster Stufe.

Inzwischen gibt es eine große Zahl verschiedener Ansätze für dieses Lernproblem, die sich hauptsächlich in der Suchrichtung im Hypothesenraum, den Generalisierungs- und Spezialisierungsoperatoren und den verwendeten nichtlogischen Beschränkungen (Bias) unterscheiden. Der *Vergleich* und die *Integration* dieser verschiedenen Ansätze war die Hauptmotivation für die Entwicklung des Systems MILES. MILES ist eine Programmierumgebung für die ILP, die neben Mechanismen zur Repräsentation und Verwaltung von Beispielen, Hintergrundwissen und Hypothesen einen Werkzeugkasten mit einem Großteil der bekannten Generalisierungs-, Spezialisierungs- und Reformulierungsoperatoren enthält. Eine generische Kontrolle erlaubt, verschiedene dieser Operatoren in einen spezifischen ILP-Algorithmus zu integrieren. In diesem Beitrag wird ein kurzer Überblick über die Repräsentation, die Operatoren und die Kontrolle von MILES gegeben.

## 2 Strukturen der Wissensbasis von MILES

Die Wissensbasis von MILES enthält positive und negative Beispiele, Hornklauseln zur Repräsentation von Hintergrundwissen und Hypothesen und eventuell Informationen über Argumenttypen von Prädikaten.

*Beispiele* werde als Fakten der Form $ex(ID, Fact, Class)$ abgelegt, wobei $ID$ ein eindeutiger Schlüssel ist, $Fact$ das Beispiel selbst und $Class$ seine Klassifikation.

*Klauseln* des Hintergrundwissens wie auch Hypothesenklauseln stehen als Fakten der Form $known(ID, Head, Body, Clist, Label, evaluation(....))$ zur Verfügung, wobei $ID$ wieder der Schlüssel ist, $Head$ und $Body$ Kopf und Rumpf der Klausel und $Label$ eine beliebige Markierung, wie z.B. der erzeugende Operator. $Clist$ enthält eine für manche Operatoren günstigere Listenrepräsentation der Klausel. Ein tiefengebundener, fortschreitend vertiefender Theorembeweiser erlaubt die Interpretation der Klauseln. Das Argument *evaluation* enthält die Auswertung der Klausel bezüglich der Beispiele, z.B. wie oft sie in Beweisen verwendet wurde.

Die Spezifikation von Argumenttypen für ein Prädikat ist optional und kann auch automatisch aus eventuell vorhandenen Beispielen erfolgen. Argumenttypen eines Prädikats $p/n$ werden als Fakt $type\_restriction(p(V_1, .., V_n), [t_1(V_1), .., t_n(V_n)])$ in der Wissensbasis abgelegt. Jeder Typ $t_i$ ist in der Wissensbasis von MILES definiert. MILES enthält neben der Methode zum Bestimmen von Argumenttypen aus Beispielen auch Prozeduren, die verschiedene Typen auf Allgemeinheit vergleichen und bei Generalisierungs- und Spezialisierungsschritten eingesetzt werden können.

*gefördert durch ESPRIT BRA 6020: Inductive Logic Programming

# 3 Die Operatoren von MILES

## 3.1 Generalisierungsoperatoren

Generalisierungsoperatoren verallgemeinern Klauseln bezüglich eines Generalisierungsmodells, das die Allgemeinheitsbeziehung zwischen Klauseln festlegt. Im Rahmen der Hornlogik erster Stufe sind drei Generalisierungsmodelle von Interesse, die die logische Implikation abschwächen: $\theta$-Subsumption [Plo70], generalisierte Subsumption [Bun88] und relative Subsumption [Plo71, MF90].

**Speziellste Generalisierungen** Für Lernverfahren ist die speziellste Generalisierung (least general generalisation, *lgg*) besonders wichtig. MILES enthält *lgg*-Operatoren für alle drei relevanten Generalisierungsmodelle: *lgg* für die $\theta$-Subsumption, *gen_msg* für die generalisierte Subsumption und *rlgg* für die relative Subsumption.

**Inverse Resolution** Inverse Resolutionsoperatoren kehren deduktive Resolutionsschritte um. Je nach Eingabe können $\mathcal{V}$- und $\mathcal{W}$-Operatoren unterschieden werden. $\mathcal{V}$-Operatoren konstruieren auf Eingabe einer Elternklausel $B$ und der Resolvente $C$ die fehlende Elternklausel. In MILES sind *Absorption* [MB88], *Identifikation* [MB88], *Saturation* [Rou91], *inverse Derivation* [Mug90] und der $G_1$-Operator [Wir89] realisiert. $\mathcal{W}$-Operatoren starten mit einer Menge von Klauseln $\{B_1, ..., B_n\}$ und konstruieren Klauseln $A$ und $\{C_1, ..., C_n\}$ so, daß $B_i$ die Resolvente von $A$ und $C_i$ ist. Da das Resolutionsliteral in $B_i$ wegfällt, führen diese Operatoren dafür *neue Prädikate* ein. MILES enthält als $\mathcal{W}$-Operatoren die *Intrakonstruktion* [MB88, Rou91] sowie den allgemeineren $G_2$-Operator [Wir89].

**Truncation** Diese Operatoren generalisieren Klauseln durch Streichen von Rumpfliteralen. Welche Rumpfliterale gestrichen werden sollen, wird heuristisch entschieden. In MILES realisierte Heuristiken sind z.B. die Verbundenheit [Rou91] oder der Ausschluß negativer Beispiele [MF90].

## 3.2 Spezialisierungsoperatoren

Die Spezialisierungsoperatoren in MILES spezialisieren eine Klausel bezüglich der $\theta$-Subsumption. Drei Operatoren, die Unifikation von Variablen, das Instantiieren von Variablen mit komplexen Termen und das Zufügen von Rumpfliteralen, sind Teil von Shapiros Verfeinerungsoperator $\rho_0$ [Sha83]. Als zusätzlicher Spezialisierungsoperator wurde in MILES die Spezialisierung durch ein neues Prädikat realisiert, wobei relevante Argumente für das neue Prädikat durch beispielbasierte Reduktion bestimmt werden [KNS92].

## 3.3 Reformulierungsoperatoren

Reformulierungsoperatoren formen die Wissensbasis äquivalent um, damit das Lernziel leichter ereicht wird. Die *Reduktion* von Klauseln bezüglich $\theta$-Subsumption [Plo70] bestimmt die kürzeste äquivalente Form einer gegebenen Klausel. Sie ist besonders wichtig für Operatoren wie *rlgg*, die Klauseln mit vielen Redundanzen erzeugen.

Das *Glätten* der Wissensbasis [Rou91] ersetzt $n$-stellige Funktoren durch $(n+1)$-stellige Prädikate und führt so zu einer äquivalenten funktionsfreien Darstellung, die Generalisierungsoperationen stark vereinfacht. Das inverse *Strukturieren* einer geglätteten Wissensbasis stellt den ursprünglichen Zustand wieder her.

## 3.4 Vorverarbeitungsoperatoren

Vorverarbeitungsoperatoren extrahieren Information, die implizit in den Beispielen gegeben ist, wie z.B. die Argumenttypen, und initialisieren die Hypothese, z.B. zu einer Menge disjunktiver Klauselköpfe, die die Unterscheidung in Basis- und rekursiven Fall eines Prädikats erlauben [STW93].

**Gegeben:** $B$, $E^{\oplus}$, $E^{\ominus}$
**Algorithmus: GENCON**
$partial\_sols := \underline{initialize}(E^{\oplus}, E^{\ominus}, B)$, $complete\_sols := \phi$
alle $PS \in partial\_sols$ als $aktiv$ markiert
while $not(\underline{Stop\_Criterion}(complete\_sols))$ do
    $PS := \underline{select}(partial\_sols)$
    if $\underline{Quality\_Criterion}(PS)$
    then $complete\_sols := complete\_sols \cup \{PS\}$
    else if $aktiv(PS)$
        then $\underline{one\_of}($ $\rightarrow partial\_sols := \underline{add}(partial\_sols, \underline{spec}(PS))$
                $\rightarrow partial\_sols := \underline{add}(partial\_sols, \underline{gen}(PS)))$
            alle $PS' \in spec(PS)(gen(PS))$ als $aktiv$ markiert
        else $partial\_sols := \underline{add}(partial\_sols, \underline{learn\_newp}(PS))$
    markiere $PS$ als $passiv$
    $partial\_sols := \underline{filter}(partial\_sols)$
**Ausgabe:** $\underline{output}(complete\_sols)$

Abbildung 1: Generische Kontrolle von MILES

### 3.5 Bewertung der Wissensbasis

Bei der Bewertung der Wissensbasis werden für jede Klausel die abgedeckten positiven und negativen Beispiele und die Beweise, in denen sie vorkommt, bestimmt. Diese Information kann benutzt werden, um z.B. die Genauigkeit einer Hypothese zu bestimmen. Zusätzlich kann geprüft werden, ob das aktuelle Programm *vollständig* und *konsistent* bezüglich der Beispiele ist. Wenn nicht, können die Prozeduren *fp* und *ip*, Implementation von Shapiros Diagnoseprozeduren [Sha83], benutzt werden, um die Fehler zu lokalisieren.

## 4 Die generische Kontrolle

Die generische Kontrolle von MILES, GENCON, erweitert den generischen Lernalgorithmus GENCOL [Rae92] um die Möglichkeit, neue Prädikate einzuführen. Die zentrale Idee von GENCON in Abbildung 1 ist, Hypothesen in *partial_sols* als *aktiv* oder *passiv* zu markieren. Aktive Programme können weiter generalisiert oder spezialisiert werden, während passive nur noch als Ausgangspunkt für das Einführen neuer Prädikate dienen. Die Funktionsweise von GENCON wird im folgenden am Beispiel regulärer Programme erklärt.

Reguläre Programme [YS91] sind auf einstellige Prädikate beschränkt und erfüllen als weitere Restriktion, daß in den Klauselrümpfen nur Variablen als Argumente vorkommen dürfen. Zusätzlich müssen sich die Kopfargumente der Klauseln für ein Prädikat im Funktor unterscheiden, und jede Kopfvariable muß durch genau ein Rumpfliteral beschränkt sein. Die Extensionen der so beschränkten Programme sind reguläre Mengen.

Die Eingabe von GENCON besteht für reguläre Programme aus einer Menge $E^{\oplus}$ positiver Beispiele und einem regulären Hintergrundwissen $B$. Negative Beispiele werden für die Induktion regulärer Programme nicht benötigt. Mit *initialize* wird *partial_sols* zu einer Menge von Klauselköpfen $\{lgg(\{p(f(...)) \in E^{\oplus}\})\}$ für jedes Prädikat $p$ und jeden Funktor $f$ in $E^{\oplus}$ initialisiert. Ziel des Induktionsprozesses ist es, für jede Variable in diesen Köpfen ein beschränkendes Rumpfliteral so zu finden, daß weiterhin alle Beispiele aus $E^{\oplus}$ abgedeckt sind. Dieses Erfolgskriterium wird in *Quality_Criterion* und *Stop_Criterion* ausgedrückt.

Mit *select* wird die spezifischste aktive Hypothese aus *partial_sols* ausgewählt. Deckt diese nicht mehr alle Beispiele in $E^{\oplus}$ ab (*one_of*), ist sie zu stark spe-

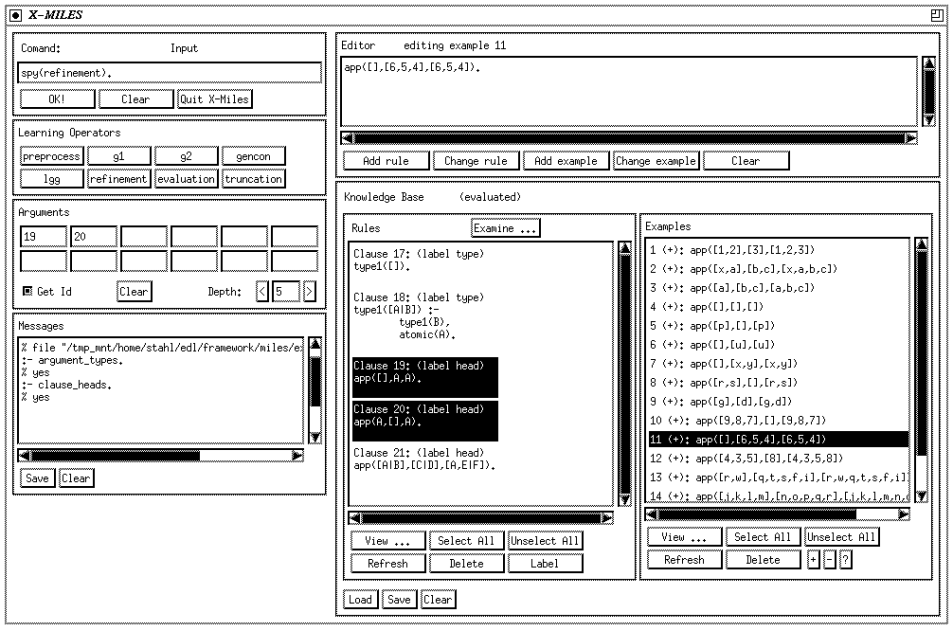


Abbildung 2: X-MILES

zialisiert. In diesem Fall wählt *gen* eine Klausel aus und generalisiert sie durch Streichen der Rumpfliterale. Im anderen Fall lokalisiert *spec* eine Klausel mit noch unbeschränkten Kopfvariablen und spezialisiert sie durch Zufügen von Rumpfliteralen, die für die Variableninstantiierungen gelten. Steht keine aktive Hypothese zur Verfügung, wählt *select* das spezifischste passive Programm zur Spezialisierung mit neuen Prädikaten in *learn_newp*. Für jede Klausel $C \in PS$ und jede unbeschränkte Variable $V \in vars(head(C)) - vars(body(C))$ wird ein Literal $newp_V(V)$ zum Rumpf von $C$ gefügt. Das Lernverfahren wird mit einer um die Instantiierungen der neuen Prädikate erweiterten Beispielmenge fortgesetzt. Dadurch kann eine reguläre Lösung des Lernproblems auch dann gefunden werden, wenn die vorhandenen Prädikate nicht zu einer vollständigen Charakterisierung ausreichen.

GENCON ist ein sehr einfacher und genereller Kontrollmechanismus, der die Integration und den Vergleich der in MILES vorhandenen Operatoren erlaubt, aber in den Parameterprozeduren einen gewissen Implementationsaufwand erfordert. In Zukunft soll dieser Aufwand minimiert werden.

# 5 Die graphische Benutzungsoberfläche

Die graphische Benutzungsoberfläche von MILES, X-MILES, ist auf der ProXT-Schnittstelle von Quintus Prolog v3.1.1 aufgebaut, die den Zugriff auf Motif und das X Toolkit erlaubt. Abbildung 2 zeigt die gesamte Oberfläche. Sie ist in sechs Bereiche aufgeteilt.

Der Bereich "Knowledge Base" stellt den aktuellen Inhalt der Wissensbasis - Regeln und Beispiele - graphisch dar und erlaubt direktes Auswählen, Entfernen und Ändern. Änderungen an Regeln und Beispielen können im Bereich "Editor" vorgenommen werden. Der Bereich "Command" erlaubt direkten Zugriff auf das darunterliegende Prolog-System, was z.B. für das Einschalten des Debuggers von Vorteil ist. Der "Message"-Bereich zeigt Systemmeldungen über Zustand und Resultate der aktuellen Operation an. Der "Learning Operators"-Bereich enthält Gruppen von Lernoperatoren, die auf die Wissensbasis angewandt werden können. Im "Arguments"-Bereich können die Argumente für diese Operatoren interaktiv aus

der Wissensbasis spezifiziert werden. Der Operatorenbereich von X-MILES kann einfach konfiguriert und den speziellen Bedürfnissen der Anwendung angepaßt werden.

# 6 Zusammenfassung

MILES ist eine prototypische Programmierumgebung für die Induktive Logische Programmierung. Es stellt eine umfassende Menge von Operationen zur Handhabung und induktiven Transformation einer Wissensbasis zur Verfügung, die einfach zu spezifischen ILP-Algorithmen kombiniert werden können.

Während der Funktionsumfang MILES als System zur raschen Entwicklung prototypischer ILP-Algorithmen besonders geeignet macht, erlaubt die einfache und übersichtliche Bedienung über X-MILES auch einen sinnvollen Einsatz in der Lehre. MILES kann mit anonymem `ftp` von `ftp.gmd.de` aus dem Verzeichnis `/MachineLearning/ILP/public/software/miles` geholt werden.

# Literatur


[Bun88] W. Buntine. Generalized subsumption and its applications to induction and redundancy. *Artificial Intelligence*, 36:149–176, 1988.

[KNS92] B. Kijsirikul, M. Numao, and M. Shimura. Discrimination-based constructive induction of logic programs. In *Proc. of the 10th National Conference on AI*, 1992.

[MB88] S. Muggleton and W. Buntine. Machine invention of first-order predicates by inverting resolution. In *Fifth International Conference on Machine Learning*. Morgan Kaufmann, 1988.

[MF90] S. Muggleton and C. Feng. Efficient induction of logic programs. In *First Conference on Algorithmic Learning Theory*, Tokyo, 1990. Ohmsha.

[Mug90] S. Muggleton. Inductive logic programming. In *First Conference on Algorithmic Learning Theory*, Tokio, October 1990. Ohmsha.

[Plo70] G. Plotkin. A note on inductive generalisation. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, Edinburgh, 1970.

[Plo71] G. Plotkin. A further note on inductive generalisation. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 6, pages 101–124. Edinburgh Univeristy Press, Edinburgh, 1971.

[Rae92] L. De Raedt. *Interactive Theory Revision: an Inductive Logic Programming Approach*. Academic Press, 1992.

[Rou91] C. Rouveirol. *ITOU: Induction de Théories en Ordre Un*. PhD thesis, Université Paris Sud, Centre d'Orsay, 1991.

[Sha83] E. Y. Shapiro. *Algorithmic Program Debugging*. MIT Press, 1983.

[STW93] I. Stahl, B. Tausend, and R. Wirth. Two methods for improving inductive logic programming systems. In *Machine Learning: ECML-93, European Conference on Machine Learning, Wien, Austria*. Springer, 1993.

[Wir89] R. Wirth. Completing logic programs by inverse resolution. In *Fourth European Working Session on Learning*. Pitman, 1989.

[YS91] E. Yardeni and E. Shapiro. A type system for logic programs. *Journal of Logic Programming*, (10):125–153, 1991.